AN EFFICIENT LATTICE ALGORITHM FOR THE LIBOR MARKET MODEL

Tim Xiao[1]

Journal of Derivatives, 19 (1) 25-40, Fall 2011

https://doi.org/10.3905/jod.2011.19.1.025

ABSTRACT

The LIBOR Market Model has become one of the most popular models for pricing interest rate products. It is commonly believed that Monte-Carlo simulation is the only viable method available for the LIBOR Market Model. In this article, however, we propose a lattice approach to price interest rate products within the LIBOR Market Model by introducing a shifted forward measure and several novel fast drift approximation methods. This model should achieve the best performance without losing much accuracy. Moreover, the calibration is almost automatic and it is simple and easy to implement. Adding this model to the valuation toolkit is actually quite useful; especially for risk management or in the case there is a need for a quick turnaround.

Key Words: LIBOR Market Model, lattice model, tree model, shifted forward measure, drift approximation, risk management, calibration, callable exotics, callable bond, callable capped floater swap, callable inverse floater swap, callable range accrual swap.

[1] Email: tim_yxiao@yahoo.com Url: https://finpricing.com/

The LIBOR Market Model (LMM) is an interest rate model based on evolving LIBOR market forward rates under a risk-neutral forward probability measure. In contrast to models that evolve the instantaneous short rates (e.g., Hull-White, Black-Karasinski models) or instantaneous forward rates (e.g., Heath-Jarrow-Morton (HJM) model), which are not directly observable in the market, the objects modeled using the LMM are market observable quantities. The explicit modeling of market forward rates allows for a natural formula for interest rate option volatility that is consistent with the market practice of using the formula of Black for caps. It is generally considered to have more desirable theoretical calibration properties than short rate or instantaneous forward rate models.

In general, it is believed that Monte Carlo simulation is the only viable numerical method available for the LMM (see Piterbarg [2003]). The Monte Carlo simulation is computationally expensive, slowly converging, and notoriously difficult to use for calculating sensitivities and hedges. Another notable weakness is its inability to determine how far the solution is from optimality in any given problem.

In this paper, we propose a lattice approach within the LMM. The model has similar accuracy to the current pricing models in the market, but is much faster. Some other merits of the model are that calibration is almost automatic and the approach is less complex and easier to implement than other current approaches.

We introduce a shifted forward measure that uses a variable substitution to shift the center of a forward rate distribution to zero. This ensures that the distribution is symmetric and can be represented by a relatively small number of discrete points. The shift transformation is the key to achieve high accuracy in relatively few discrete finite nodes. In addition, we present several fast and novel drift approximation approaches. Other concepts used in the model are probability distribution structure

exploitation, numerical integration and the long jump technique (we only position nodes at times when decisions need to be made).

This model is actually quite useful for risk management because normally full-revaluations of an entire portfolio under hundreds of thousands of different future scenarios are required for a short time window (see FinPricing (2011)). Without an efficient algorithm, one cannot properly capture and manage the risk exposed by the portfolio.

The rest of this paper is organized as follows: The LMM is discussed in Section I. In Section II, the lattice model is elaborated. The calibration is presented in Section III. The numerical implementation is detailed in Section IV, which will enhance the reader's understanding of the model and its practical implementation. The conclusions are provided in Section V.

I. LIBOR MARKET MODEL

Let $(\Omega, \mathcal{F}, \{\mathcal{F}_t\}_{t\geq 0}, \mathcal{P})$ be a filtered probability space satisfying the usual conditions, where Ω denotes a sample space, $\mathcal{F}$ denotes a σ-algebra, $\mathcal{P}$ denotes a probability measure, and $\{\mathcal{F}_t\}_{t\geq 0}$ denotes a filtration. Consider an increasing maturity structure $0 = T_0 < T_1 < \ldots < T_N$ from which expiry-maturity pairs of dates (T_{k-1}, T_k) for a family of spanning forward rates are taken. For any time $t \leq T_{k-1}$, we define a right-continuous mapping function $n(t)$ by $T_{n(t)-1} \leq t < T_{n(t)}$. The simply compounded forward rate reset at *t* for forward period (T_{k-1}, T_k) is defined by

$$F_k(t) \coloneqq F(t; T_{k-1}, T_k) = \frac{1}{\delta_k}\left(\frac{P(t,T_{k-1})}{P(t,T_k)} - 1\right) \tag{1}$$

where $P(t,T)$ denotes the time *t* price of a zero-coupon bond maturing at time *T* and $\delta_k \coloneqq \delta(T_{k-1}, T_k)$ is the accrual factor or day count fraction for period (T_{k-1}, T_k).

Inverting this relationship (1), we can express a zero coupon bond price in terms of forward rates as:

$$P(t,T_k) = P(t,T_{n(t)}) \prod_{j=n(t)}^{k} \frac{1}{1+\delta_j F_j(t)} \tag{2}$$

LIBOR Market Model Dynamics

Consider a zero coupon bond numeraire $P(\bullet,T_i)$ whose maturity coincides with the maturity of the forward rate. The measure Q^i associated with $P(\bullet,T_i)$ is called T_i forward measure. Terminal measure Q^N is a forward measure where the maturity of the bond numeraire $P(\bullet,T_N)$ matches the terminal date T_N.

For brevity, we discuss the one-factor LMM only. The one-factor LMM (Brace et al. [1997]) under forward measure Q^i can be expressed as

$$\text{If } i<k,\ t \le T_i,\quad dF_k(t) = \sigma_k(t)F_k(t)\sum_{j=i+1}^{k} \frac{\delta_j \sigma_j(t) F_j(t)}{1+\delta_j F_j(t)} dt + \sigma_k(t)F_k(t)dX_t \tag{3a}$$

$$\text{If } i=k,\ t \le T_{k-1},\quad dF_k(t) = \sigma_k(t)F_k(t)dX_t \tag{3b}$$

$$\text{If } i>k,\ t \le T_{k-1},\quad dF_k(t) = -\sigma_k(t)F_k(t)\sum_{j=k+1}^{i} \frac{\delta_j \sigma_j(t) F_j(t)}{1+\delta_j F_j(t)} dt + \sigma_k(t)F_k(t)dX_t \tag{3c}$$

where X_t is a Brownian motion.

There is no requirement for what kind of instantaneous volatility structure should be chosen during the life of the caplet. All that is required is (see Hull-White [2000]):

$$(\bar{\sigma}_k)^2 := \left(\bar{\sigma}_k(T_{k-1},\mathrm{K})\right)^2 = \frac{1}{T_{k-1}} \int_0^{T_{k-1}} \sigma_k^2(u)du \tag{4}$$

where $\bar{\sigma}_k$ denotes the market Black caplet volatility and K denotes the strike. Given this equation, it is obviously not possible to uniquely pin down the instantaneous volatility function. In fact, this specification allows an infinite number of choices. People

often assume that a forward rate has a piecewise constant instantaneous volatility. Here we choose the forward rate $F_k(t)$ has constant instantaneous volatility regardless of *t* (see Brigo-Mercurio [2006]).

Shifted Forward Measure

The $F_k(t)$ is a Martingale or driftless under its own measure Q^k. The solution to equation (3b) can be expressed as

$$F_k(t) = F_k(0)\exp\left(-\frac{1}{2}\int_0^t \sigma_k(s)^2\,ds + \int_0^t \sigma_k(s)\,dX_s\right) \tag{5}$$

where $F_k(0) = F(0;T_{k-1},T_k)$ is the current (spot) forward rate. Under the volatility assumption described above, equation (5) can be further expressed as

$$F_k(t) = F_k(0)\exp\left(-\frac{\sigma_k^{\,2}}{2}t + \sigma_k X_t\right) \tag{6}$$

Alternatively, we can reach the same Martingale conclusion by directly deriving the expectation of the forward rate (6); that is

$$\begin{aligned} E_0\left(F_k(t)\right) &= F_k(0)\frac{1}{\sqrt{2\pi t}}\int_{-\infty}^{\infty}\exp\left(-\frac{\sigma_k^{\,2}}{2}t + \sigma_k X_t\right)\exp\left(-\frac{X_t^{\,2}}{2t}\right)dX_t \\ &= F_k(0)\frac{1}{\sqrt{2\pi t}}\int_{-\infty}^{\infty}\exp\left(-\frac{(X_t - t\sigma_k)^2}{2t}\right)dX_t = F_k(0)\frac{1}{\sqrt{2\pi t}}\int_{-\infty}^{\infty}\exp\left(-\frac{Y_t^{\,2}}{2t}\right)dY_t = F_k(0) \end{aligned} \tag{7}$$

where X_t, Y_t are both Brownian motions with a normal distribution (0, *t*) at time *t*, $E_t(\bullet) \coloneqq E(\bullet \mid \mathcal{F}_t)$ is the expectation conditional on the $\mathcal{F}_t$, and the variable substitution used for derivation is

$$Y_t = X_t - t\sigma_k \tag{8}$$

This variable substitution that ensures that the distribution is centered on zero and symmetry is *the key to achieve high accuracy* when we express the LMM in discrete finite form and use numerical integration to calculate the expectation. As a matter of

fact, without this linear transformation, a lattice method in the LMM either does not exist or introduces too much error for longer maturities.

After applying this variable substitution (8), equation (6) can be expressed as

$$F_k(t) = F_k(0)\exp\left(-\frac{\sigma_k^{\,2}}{2}t + \sigma_k X_t\right) = F_k(0)\exp\left(\frac{\sigma_k^{\,2}}{2}t + \sigma_k Y_t\right) \tag{9}$$

Since the LMM models the complete forward curve directly, it is essential to bring everything under a common measure. The terminal measure is a good choice for this purpose, although this is by no means the only choice. The forward rate dynamic under terminal measure Q^N is given by

$$dF_k(t) = -\sigma_k F_k(t)\sum_{j=k+1}^{N}\frac{\delta_j\sigma_j F_j(t)}{1+\delta_j F_j(t)}dt + \sigma_k F_k(t)dX_t \tag{10}$$

The solution to equation (10) can be expressed as

$$F_k(t) = F_k(0)\exp\left(\mu_k(t) - \int_0^t\frac{\sigma_k^{\,2}}{2}ds + \int_0^t\sigma_k dX_s\right) = F_k(0)\exp\left(\mu_k(t) - \frac{\sigma_k^{\,2}}{2}t + \sigma_k X_t\right) \tag{11a}$$

where the drift is given by

$$\mu_k(t) = -\int_0^t\sum_{j=k+1}^{N}\psi_j(s)\sigma_k\sigma_j ds = -\int_0^t\sum_{j=k+1}^{N}\frac{\delta_j F_j(s)}{1+\delta_j F_j(s)}\sigma_k\sigma_j ds \tag{11b}$$

where $\psi_j(s) = \delta_j F_j(s)/\left[1+\delta_j F_j(s)\right]$ is the drift term.

Applying (8) to (11a), we have the forward rate dynamic under the shifted terminal measure as

$$F_k(t) = F_k(0)\exp\left(\mu_k(t) + \frac{\sigma_k^{\,2}}{2}t + \sigma_k Y_t\right) \tag{12}$$

Drift Approximation

Under terminal measure, the drifts of forward rate dynamics are state-dependent, which gives rise to sufficiently complicated non-lognormal distributions.

This means that an explicit analytic solution to the forward rate stochastic differential equations cannot be obtained. Therefore, most work on the topic has focused on ways to approximate the drift, which is the fundamental trickiness in implementing the Market Model.

Our model works backwards recursively from forward rate *N* down to forward rate *k*. The N-th forward rate $F_N(t)$ without drift can be determined exactly. By the time it takes to calculate the k-th forward rate $F_k(t)$, all forward rates from $F_{k+1}(t)$ to $F_N(t)$ at time t are already known. Therefore, the drift calculation (11b) is to estimate the integrals containing forward rate dynamics $F_j(s)$, for *j=k+1,…,N*, with known beginning and end points given by $F_j(0)$ and $F_j(t)$. For completeness, we list all possible solutions below.

Frozen Drift (FD). Replace the random forward rates in the drift by their deterministic initial values, i.e.,

$$\mu_k(t) = -\int_0^t \sum_{j=k+1}^{N} \frac{\delta_j F_j(s)}{1+\delta_j F_j(s)} \sigma_k \sigma_j ds \approx -\sum_{j=k+1}^{N} \frac{\delta_j F_j(0)}{1+\delta_j F_j(0)} \sigma_k \sigma_j t \qquad (13)$$

Arithmetic Average of the Forward Rates (AAFR). Apply the midpoint rule (rectangle rule) to the random forward rates in the drift, i.e.,

$$\mu_k(t) \approx -\sum_{j=k+1}^{N} \frac{\delta_j \frac{1}{2}\left(F_j(0)+F_j(t)\right)}{1+\delta_j \frac{1}{2}\left(F_j(0)+F_j(t)\right)} \sigma_k \sigma_j t \qquad (14)$$

Arithmetic Average of the Drift Terms (AADT). Apply the midpoint rule to the random drift terms, i.e.,

$$\mu_k(t) \approx -\sum_{j=k+1}^{N} \frac{1}{2}\left(\frac{\delta_j F_j(0)}{1+\delta_j F_j(0)} + \frac{\delta_j F_j(t)}{1+\delta_j F_j(t)} \right) \sigma_j \sigma_k t \qquad (15)$$

Geometric Average of the Forward Rates (GAFR). Replace the random forward rates in the drift by their geometric averages, i.e.,

$$\mu_k(t) \approx -\sum_{j=k+1}^{N} \frac{\delta_j \sqrt{F_j(0) \times F_j(t)}}{1+\delta_j \sqrt{F_j(0) \times F_j(t)}} \sigma_j \sigma_k t \tag{16}$$

Geometric Average of the Drift Terms (GADT). Replace the random drift terms by their geometric averages, i.e.,

$$\mu_k(t) \approx -\sum_{j=k+1}^{N} \sqrt{\frac{\delta_j F_j(0)}{1+\delta_j F_j(0)} \times \frac{\delta_j F_j(t)}{1+\delta_j F_j(t)}} \sigma_j \sigma_k t \tag{17}$$

Conditional Expectation of the Forward Rate (CEFR). In addition to the two endpoints, we can further enhance our estimate based on the dynamics of the forward rates. The forward rate $F_j(s)$ follows the dynamic (9) (The drift term is ignored). We can derive the expectation of the forward rate conditional on the two endpoints and replace the random forward rate in the drift by the conditional expectation of the forward rate.

Proposition 1*. Assume the forward rate $F_j(s)$ follows the dynamic (9), with the two known endpoints given by $F_j(0)$ and $F_j(t)$. Based on the conditional expectation of the forward rate $F_j(s)$, the drift of $F_k(t)$ can be expressed as*

$$\mu_k(t) \approx -\sum_{j=k+1}^{N} \int_0^t \frac{\delta_j E_0[F_j(s)|_{F_j(0),F_j(t)}]}{1+\delta_j E_0[F_j(s)|_{F_j(0),F_j(t)}]} \sigma_j \sigma_k ds \tag{18a}$$

where the conditional expectation of the forward rate is given by

$$E_0[F_j(s)|_{F_j(0),F_j(t)}] = F_j(0)\left(\frac{F_j(t)}{F_j(0)}\right)^{\frac{s}{t}} \exp\left(\frac{\sigma_j^2 s(t-s)}{2t}\right) \tag{18b}$$

Proof. See Appendix A.

Conditional Expectation of the Drift Term (CEDT). Similarly, we can calculate the conditional expectation of the drift term and replace the random drift term by the conditional expectation.

Proposition 2*. Assume the forward rate* $F_j(s)$ *follows the dynamic (9), with the two known endpoints given by* $F_j(0)$ *and* $F_j(t)$*. Based on the conditional expectation of the drift term* ψ_j*, the drift of* $F_k(t)$ *can be expressed as*

$$\mu_k(t) \approx -\sum_{j=k+1}^{N} \int_0^t E_0\left(\frac{\delta_j F_j(s)}{1+\delta_j F_j(s)}\bigg|_{F_j(0),F_j(t)} \right) \sigma_j \sigma_k ds \qquad (19a)$$

where the conditional expectation of the drift term is given by

$$E_0\left(\psi_j(s)|_{F_j(0),F_j(t)}\right) = E_0\left(\frac{\delta_j F_j(s)}{1+\delta_j F_j(s)}\bigg|_{F_j(0),F_j(t)} \right) = 1 - \frac{1+\nu_{Cj}(s)/\mu_{Cj}^2(s)}{\mu_{Cj}(s)} \qquad (19b)$$

$$\mu_{Cj}(s) = 1+\delta_j F_j(0)\left(\frac{F_j(t)}{F_j(0)}\right)^{\frac{s}{t}} \exp\left(\frac{\sigma_j^2 s(t-s)}{2t}\right) \qquad (19c)$$

$$\nu_{Cj}(s) = \delta_j^2 F_j^2(0)\left(\frac{F_j(t)}{F_j(0)}\right)^{\frac{2s}{t}} \left(\exp\left(\frac{\sigma_j^2 s(t-s)}{t}\right)-1\right)\exp\left(\frac{\sigma_j^2 s(t-s)}{t}\right) \qquad (19d)$$

Proof. See Appendix A.

The accuracy and performance of these drift approximation methods are discussed in section IV.

II. THE LATTICE PROCEDURE IN THE LMM

The "lattice" is the generic term for any graph we build for the pricing of financial products. Each lattice is a layered graph that attempts to transform a continuous-time and continuous-space underlying process into a discrete-time and discrete-space process, where the nodes at each level represent the possible values of the underlying process in that period.

There are two primary types of lattices for pricing financial products: tree lattices and grid lattices (or rectangular lattices or Markov chain lattices). The tree lattices, e.g., traditional binomial tree, assume that the underlying process has two

possible outcomes at each stage. In contrast with the binomial tree lattice, the grid lattices (see Amin [1993], Gandhi-Hunt [1997], Martzoukos-Trigeorgis [2002], Hagan [2005], and Das [2011]) shown in Exhibit 1, which permit the underlying process to change by multiple states, are built in a rectangular finite difference grid (not to be confused with finite difference numerical methods for solving partial differential equations). The grid lattices are more realistic and convenient for the implementation of a Markov chain solution.

This article presents a grid lattice model for the LMM. To illustrate the lattice algorithm, we use a callable exotic as an example. Callable exotics are a class of interest rate derivatives that have Bermudan style provisions that allow for early exercise into various underlying interest rate products. In general, a callable exotic can be decomposed into an underlying instrument and an embedded Bermudan option.

We will simplify some of the definitions of the universe of instruments we will be dealing with for brevity. Assume the payoff of a generic underlying instrument is a stream of payments $Z_i = \delta_i\left[F_i(T_{i-1}) - C_i\right]$ for *i=1,…,N*, where C_i is the structured coupon. The callable exotic is a Bermudan style option to enter the underlying instrument on any of a sequence of notification dates $t_1^{ex}, t_2^{ex}, \ldots, t_M^{ex}$. For any notification date $t = t_j^{ex}$, we define a right-continuous mapping function $n(t)$ by $T_{n(t)-1} \le t < T_{n(t)}$. If the option is exercised at *t*, the reduced price of the underlying instrument, from the structured coupon payer's perspective, is given by

$$\tilde{I}(t) := \frac{I(t)}{P(t,T_N)} = \sum_{i=n(t)}^{N} E_t\left(\frac{Z_i}{P(T_i,T_N)}\right) = \sum_{i=n(t)}^{N} E_t\left(\frac{\delta_i\left(F_i(T_{i-1}) - C_i\right)}{P(T_i,T_N)}\right) \tag{20}$$

where the ratio $\tilde{I}(t)$ is usually called the reduced value of the underlying instrument or the reduced exercise value or the reduced intrinsic value.

Lattice approaches are ideal for pricing early exercise products, given their "backward-in-time" nature. Bermudan pricing is usually done by building a lattice to

carry out a dynamic programming calculation via backward induction and is standard. The lattice model described below also uses backward induction but exploits the Gaussian structure to gain extra efficiencies.

First we need to create the lattice. The random process we are going to model in the lattice is the LMM (12). Unlike traditional trees, we only position nodes at the determination dates (the payment and exercise dates). At each determination date, the continuous-time stochastic equation (12) shall be discretized into a discrete-time scheme. Such discretized schemes basically convert the Brownian motion into discrete variables. There is no restriction on discretization schemes. At any determination date *t*, for instance, we discretize the Brownian motion to be equally spaced as a grid of nodes $y_{i,t}$, for *i = 1,…,* S_t. The number of nodes S_t and the space between nodes $\varphi_t = y_{i,t} - y_{i-1,t}$ at each determination date can vary depending on the length of time and the accuracy requirement. The nodes should cover a certain number of standard deviations of the Gaussian distribution to guarantee a certain level of accuracy. We have the discrete form of the forward rate as

$$F_k(t; y_{i,t}) = F_k(0) \exp\left(\mu_k(t, y_{i,t}) + \frac{\sigma_k^2}{2} t + \sigma_k y_{i,t} \right) \tag{21}$$

The zero-coupon bond (2) can be expressed in discrete form as

$$P(t, T_k; y_{i,t}) = P(t, T_{n(t)}; y_{i,t}) \prod_{j=n(t)}^{k} \frac{1}{1 + \delta_j F_j(t; y_{i,t})} \tag{22}$$

We now have expressions for the forward rate (21) and discount bond (22), conditional on being in the state $y_{i,t}$ at time *t*, and from these we can perform valuation for the underlying instrument.

At the maturity date, the value of the underlying instrument is equal to the payoff, i.e.,

$$I(T_N, y_{i,T_N}) = Z_N(y_{i,T_N}) \tag{23}$$

The underlying state process X_t in the LMM (11) is a Brownian motion. The transition probability density from state ($x_{i,t}$, t) to state ($x_{j,T}$, T) is given by

$$p(x_{i,t},t;x_{j,T},T)=\frac{1}{\sqrt{2\pi(T-t)}}\exp\left[-\frac{(x_{j,T}-x_{i,t})^2}{2(T-t)}\right] \tag{24}$$

Applying the variable substitution (8), equation (24) can be expressed as

$$p(y_{i,t},t;y_{j,T},T)=\frac{1}{\sqrt{2\pi(T-t)}}\exp\left[-\frac{(y_{j,T}-y_{i,t}+\sigma_T T-\sigma_t t)^2}{2(T-t)}\right] \tag{25}$$

Equation (20) can be further expressed as a conditional value on any state ($y_{i,t}$, t) as:

$$\frac{I(t;y_{i,t})}{P(t,T_N;y_{i,t})}=\sum_{j=n(t)}^{N}\frac{1}{\sqrt{2\pi(T_j-t)}}\int\frac{Z_j(y_{T_j})}{P(T_j,T_N;y_{T_j})}\exp\left[-\frac{(y_{T_j}-y_{i,t}+\sigma_{T_j}T_j-\sigma_t t)^2}{2(T_j-t)}\right]dy_{T_j} \tag{26}$$

This is a convolution integral. Some fast integration algorithms, e.g., Cubic Spline Integration, Fast Fourier Transform (FFT), etc., can be used for optimization. We use the Trapezoidal Rule Integration in this paper for ease of illustration.

Incomplete information handling. Convolution is widely used in Electrical Engineering, particularly in signal processing. The important part is that the far left and far right parts of the output are based on incomplete information. Any models that try to compute the transition values using integration will be inaccurate if this problem is not solved, especially for longer maturities and multiple exercise dates. Our solution is to extend the input nodes by padding the far end values on each side and only take the original range of the output nodes.

Next, we determine the option values in each final notification node. On the last exercise date, if we have not already exercised, the reduced option value in any state $y_{i,M}$ is given by

$$\frac{V(t_M^{ex},y_{i,M})}{P(t_M^{ex},T_N;y_{i,M})}=\max\left(\frac{I(t_M^{ex};y_{i,M})}{P(t_M^{ex},T_N;y_{i,M})},0\right) \tag{27}$$

Then, we conduct the backward induction process that is performed by iteratively rolling back a series of long jumps from the final exercise date t_M^{ex} across notification dates and exercise opportunities until we reach the valuation date. Assume that in the previous rollback step t_j^{ex}, we calculated the reduced option value: $V(t_j^{ex}, y_{i,j}) / P(t_j^{ex}, T_N; y_{i,j})$. Now, we go to t_{j-1}^{ex}. The reduced option value at t_{j-1}^{ex} is

$$\frac{V(t_{j-1}^{ex}, y_{i,j-1})}{P(t_{j-1}^{ex}, T_N; y_{i,j-1})} = \max\left\{ \frac{I(t_{j-1}^{ex}, y_{i,j-1})}{P(t_{j-1}^{ex}, T_N; y_{i,j-1})}, \frac{V^c(t_{j-1}^{ex}, y_{i,j-1})}{P(t_{j-1}^{ex}, T_N; y_{i,j-1})} \right\} \tag{28a}$$

where the reduced continuation value is given by

$$\frac{V^c(t_{j-1}^{ex}, y_{i,j-1})}{P(t_{j-1}^{ex}, T_N; y_{i,j-1})} = \frac{1}{\sqrt{2\pi(t_j^{ex} - t_{j-1}^{ex})}} \int \frac{V(t_j^{ex}, y_j)}{P(t_j^{ex}, T_N; y_j)} \exp\left[-\frac{(y_j - y_{i,j-1} + \sigma_j t_j^{ex} - \sigma_{j-1} t_{j-1}^{ex})^2}{2(t_j^{ex} - t_{j-1}^{ex})} \right] dy_j \tag{28b}$$

We repeat the rollback procedure and eventually work our way through the first exercise date. Then the present value of the Bermudan option is found by a final integration given by

$$pv_{Bermudan}(0) = P(0, T_N) \frac{1}{\sqrt{2\pi t_1^{ex}}} \int \frac{V(t_1^{ex}, y_1)}{P(t_1^{ex}, T_N; y_1)} \exp\left[-\frac{\left(y_1 + \sigma_1 t_1^{ex}\right)^2}{2t_1^{ex}} \right] dy_1 \tag{29}$$

The present value or the price of the callable exotic from the coupon payer's perspective is:

$$pv_{payer}(0) = pv_{Bermudan}(0) - pv_{underly_instrument}(0) \tag{30}$$

This framework can be used to price any interest rate products in the LMM setting and can be easily extended to the Swap Market Model (SMM).

III. Calibration

First, if we choose the LMM as the central model, we need to price interest rate derivatives that depend on either or both of cap and swaption markets. Second, we will undoubtedly use various swaptions to hedge a callable exotic. It is a reasonable

expectation that the calibrated model we intend to use to price our exotic, will at least correctly price the market instruments that we intend to hedge with. Therefore, in an exotic derivative pricing situation, recovery of both cap and swaption markets might be desired.

The calibration of the LMM to caplet prices is quite straightforward. However, it is very difficult, if not impossible, to perfectly recover both cap and swaption markets. Fortunately for the LMM, there also exist extremely accurate approximate formulas for swaptions implied volatility, e.g., Rebonato's formula.

We introduced a parameter θ and set $\sigma_i = \theta\hat{\sigma}_i$ where $\hat{\sigma}_i$ denotes the market Black caplet volatility. One can choose different θ for different σ_i. For simplicity we describe one θ situation here. By choosing $\theta = 1$, we have perfectly calibrated the LMM to the caplet prices in the market. However, our goal is to select a θ to minimize the sum of the squared differences of the volatilities derived from the market and the volatilities implied by our model for both caps and swaptions combined.

In the optimization, we use Rebonato's formula for an efficient expression of the model swaption volatilities, given by

$$\begin{aligned}\left(\upsilon_{\alpha,\beta}^{LMM}\right)^2 &= \frac{1}{T_\alpha}\sum_{i,j=\alpha+1}^{\beta}\frac{w_i(0)w_j(0)F_i(0)F_j(0)\rho_{ij}}{S_{\alpha,\beta}(0)^2}\int_0^{T_\alpha}\sigma_i(t)\sigma_j(t)dt \\ &= \sum_{i,j=\alpha+1}^{\beta}\frac{w_i(0)w_j(0)F_i(0)F_j(0)\hat{\sigma}_j\hat{\sigma}_j\theta^2}{S_{\alpha,\beta}(0)^2} = \theta^2\left(\upsilon_{\sigma,\beta}^{\text{Rebonato}}\right)^2\end{aligned} \tag{31a}$$

where ρ_{ij} =1 under one-factor LMM. The swap rate $S_{\alpha,\beta}(0)$ is given by

$$S_{\alpha,\beta}(0) = \sum_{i=\alpha+1}^{\beta} w_i(0)F_i(0) \tag{31b}$$

$$w_i(0) = \frac{\delta_i\prod_{j=\alpha+1}^{\beta}\left(1+\delta_j F_j(0)\right)^{-1}}{\sum_{k=\alpha+1}^{\beta}\delta_k\prod_{j=\alpha+1}^{k}\left(1+\delta_j F_j(0)\right)^{-1}} \tag{31c}$$

Assume the calibration containing M caplets and G swaptions. The error minimization is given by

$$\min\left|\sum_{i=1}^{M}\left(\theta\widehat{\sigma}_i-\widehat{\sigma}_i\right)^2+\sum_{j=1}^{G}\left(\theta\upsilon_{\alpha+j,N}^{\text{Rebonato}}-\sigma_{\alpha+j,N}^{swn}\right)^2\right| \tag{32}$$

where $\sigma_{\alpha+j,N}^{swn}$ denotes the market Black swaption volatility. The optimization can be found at a stationary point where the first derivative is zero; that is,

$$\theta=\frac{\sum_{i=1}^{M}\widehat{\sigma}_i+\sum_{j=1}^{G}\sigma_{\alpha+j,N}^{swn}}{\sum_{i=1}^{M}\widehat{\sigma}_i+\sum_{j=1}^{G}\upsilon_{\alpha+j,N}^{\text{Rebonato}}} \tag{33}$$

In terms of forward volatilities, we use the time-homogeneity assumption of the volatility structure, where a forward volatility for an option is the same or close to the spot volatility of the option with the same time to expiry. The time-homogeneous volatility structure can avoid non-stationary behavior.

In the LMM, forward swap rates are generally not lognormal. Such deviation from the lognormal paradigm however turns out to be extremely small. Rebonato [1999] shows that the pricing errors of swaptions caused by the lognormal approximation are well within the market bid/ask spread. For most short maturity interest rate products, we can use the lattice model without calibration (33). However, for longer maturity or deeply in the money (ITM) or out of the money (OTM) exotics we may need to use the calibration and even some specific skew/smile adjustment techniques to achieve high accuracy.

IV. NUMERICAL IMPLEMENTATION

In this section, we will elaborate on more details of the implementation. We will start with a simple callable bond for the purpose of an easy illustration and then move on to some typical callable exotics, e.g., callable capped floater swap and callable range accrual swap. The reader should be able to implement and replicate the model after reading this section.

Callable Bond

A callable bond is a bond with an option that allows the issuer to retain the privilege of redeeming the bond at some points before the bond reaches the maturity date. For ease of illustration, we choose a very simple callable bond with a one-year maturity, a quarterly payment frequency, a $100 principal amount (*A*), and a 4% annual coupon rate (the quarterly coupon $C=1$). The call dates are 6 months, 9 months, and 12 months. The call price (*H*) is 100% of the principal. The bond spread (ξ) is 0.002. Let the valuation date be 0. A detailed description of the callable bond and current (spot) market data is shown in Exhibit 2.

For a short-term maturity callable bond, our lattice model can reach high accuracy even without calibration (33) and incomplete information handling. Therefore, we set $\theta=1$ and $\sigma_i=\hat{\sigma}_i$. The valuation procedure for a callable bond consists of 4 steps:

Step 1: Create the lattice. Based on the long jump technique, we position nodes only at the determination (payment/exercise) dates. The number of nodes and the space between nodes at each determination date may vary depending on the length of time and the accuracy requirement. To simplify the illustration, we choose the same settings across the lattice, with a grid space (space between nodes) $\varphi=1/2$, and a number of nodes *S*=7. It covers $\varphi(S-1)=3$ standard deviations for a standard normal distribution. The nodes are equally spaced and symmetric, as shown in Exhibit 3.

Step 2: Find the option value at each final node. At the final maturity date T_4 , the payoff of the callable bond in any state y_i is given by

$$V_{i,4} := V(T_4, y_i) = \min\left(H, A+C\right) \tag{34}$$

where *A* denotes the principal amount, *C* denotes the bond coupon, and *H* denotes the call price. The option values at the maturity are equal to the payoffs as shown in Exhibit 3.

Step 3: Find the option value at earlier nodes. Let us go to the penultimate notification date T_3. The option value in any state y_i is given by

$$V_{i,3} := V(T_3, y_i) = \min\left(H, V_{i,3}^c + C\right) \tag{35}$$

Equation (35) can be further expressed in the form of reduced value as

$$\tilde{V}_{i,3} := \frac{V_{i,3}}{P(T_3,T_4;y_i)} = \min\left(\frac{H}{P(T_3,T_4;y_i)}, \frac{V_{i,3}^c + C}{P(T_3,T_4;y_i)}\right) \tag{36a}$$

where $V_{i,3}^C / P(T_3,T_4;y_i)$ denotes the reduced continuation value in state y_i at T_3 given by

$$\begin{aligned}\frac{V_{i,3}^c}{P(T_3,T_4;y_i)} &= \frac{\exp\left(-\xi(T_4-T_3)\right)}{\sqrt{2\pi(T_4-T_3)}}\int V(T_4,Y)\exp\left[-\frac{(Y-y_i+\sigma_4 T_4-\sigma_3 T_3)^2}{2(T_4-T_3)}\right]dY \\ &= \frac{\exp\left(-\xi(T_4-T_3)\right)}{\sqrt{2\pi(T_4-T_3)}}\frac{\varphi}{2}\sum_{j=2}^{7}\left\{V(T_4,y_j)\exp\left[-\frac{(y_j-y_i+\sigma_4 T_4-\sigma_3 T_3)^2}{2(T_4-T_3)}\right]\right. \\ &\quad \left. + V(T_4,y_{j-1})\exp\left[-\frac{(y_{j-1}-y_i+\sigma_4 T_4-\sigma_3 T_3)^2}{2(T_4-T_3)}\right]\right\}\end{aligned} \tag{36b}$$

where ξ denotes the bond spread. Similarly we can compute the reduced callable bond values at T_2. All intermediate reduced values are shown in Exhibit 3.

Step 4: Compute the final integration. The final integral at valuation date 0 is calculated as

$$\begin{aligned}\overline{V}(0) &= P(0,T_4)\frac{\exp\left(-\xi T_2\right)}{\sqrt{2\pi T_2}}\int \frac{V(T_2,Y)}{P(T_2,T_4;Y)}\exp\left[-\frac{(Y+\sigma_2 T_2)^2}{2T_2}\right]dY \\ &= P(0,T_4)\frac{\exp\left(-\xi T_2\right)}{\sqrt{2\pi T_2}}\frac{\varphi}{2}\sum_{j=2}^{7}\left\{\frac{V(T_2,y_j)}{P(T_2,T_4;y_j)}\exp\left[-\frac{(y_j+\sigma_2 T_2)^2}{2T_2}\right]\right. \\ &\quad \left. + \frac{V(T_2,y_{j-1})}{P(T_2,T_4;y_{j-1})}\exp\left[-\frac{(y_{j-1}+\sigma_2 T_2)^2}{2T_2}\right]\right\} = 80.399\end{aligned} \tag{37}$$

Moreover, we need to add the present value of the coupon at T_1 into the final price. The final callable bond value is given by

$$V(0) = \overline{V}(0) + \exp(-\xi T_1) P(0, T_1) C = 81.398 \tag{38}$$

The pseudo-code is supplied in Appendix B for the implementation program. The convergence results shown in Exhibit 4 indicate what occurs for a given grid space φ when we increase the number of nodes *S*. The speed of convergence is very fast, ensuring that a small number of grids are sufficient. All calculations are converged to 100.7518. One sanity check is that the callable bond price should be close to the straight bond price if the call prices become very high. Both of them are computed as 103.3536.

Callable capped floater swap

A callable capped floater swap has two legs: a regular floating leg and a structured coupon leg. The structured coupon rate of the j-th period (T_{j-1}, T_j) is given by

$$C_j = A_j \delta_j \max\{\min[\eta_j + \lambda_j F_j(T_{j-1}), K_j^C], K_j^F\} \tag{39}$$

where A_j is the notional amount, K_j^C is the rate cap, K_j^F is the rate floor, η_j is the spread and λ_j is the scale factor. For $\lambda_j > 0$, it is called a callable capped floater swap. For $\lambda_j < 0$, it is called a callable inverse floater swap.

We choose a real middle life trade with more than 10 years remaining in its lifetime. The floating leg has a quarterly payment frequency with step-down notionals and step-up spreads. The structured coupon leg has a semi-annually payment frequency with varying notionals, spreads, scales, rate caps, and rate floors. The call schedule is semi-annual.

Callable range accrual swap

A callable range accrual swap has two legs: a regular floating leg and a structured coupon leg. The structured coupon rate of the j-th period (T_{j-1}, T_j) is given by

$$C_j = \sum_{t_i = T_{j-1}+1}^{T_j} \frac{A_j \delta_j R I_i}{M_j} \tag{40a}$$

where

$$I_i = \begin{cases} 1 & if\ K_{j\min} \le F(t_i; t_i, t_i + \varsigma) \le K_{j\max} \\ 0 & otherwise \end{cases} \tag{40b}$$

where *R* is the fixed rate, $K_{j_{\min}}$ and $K_{j_{\max}}$ are the accrual range of the j-th period, $F(t_i; t_i, t_i + \varsigma)$ is the LIBOR rate, ς is the range accrual index term, M_j is the total number of the business days in the j-th period.

We choose a real 10 years maturity trade. The floating leg has a quarterly payment frequency and the structured coupon leg has a semi-annually payment frequency with varying accrual ranges. It starts with the first call opportunity being in 3 years from inception, and then every year until the last possibility being 9 years from inception. The range accrual index term is 6 months.

The lattice implementation procedure for a callable capped floater swap or a callable range accrual swap is quite similar to the one for a callable bond except the valuation for the underlying instrument.

The convergence diagrams of pricing calculations are shown in Exhibits 5 and 6. Each curve in the diagrams represents the convergence behavior for a given grid space as nodes are increased. All of the lattice results are well converged. If the grid space is smaller, the algorithm has better convergence accuracy but a slower convergence rate, and vice verse.

We benchmarked our model under different drift approximation methods with several standard market approaches, e.g., the regression-based Monte Carlo in the full LMM and the HJM trinomial tree. The model comparisons for the accuracy and speed are shown in Exhibits 7 and 8. With regards to accuracy, as expected, the FD performs very badly. AAFR and GAFR do a little better but errors go in different directions. The same conclusions can be drawn for AADT and GADT. Both CEFR and CEDT are the best. In terms of CPU times, FD, AAFR, AADT, GAFR and GADT are the same. But CEFR and CEDT are slower, especially in the callable range accrual swap case.

V. CONCLUSION

In this paper, we proposed a lattice model in the LMM to price interest rate products. Conclusions can be drawn, supported by the previous sections. First, the model is quite stable. The fast convergence behavior requires fewer discretization nodes. Second, this model has almost equivalent accuracy to the current pricing models in the market. Third, the implementation of the model is relatively easy. The calibration is very simple and straightforward. Finally, the performance of the model is probably the best among all known approaches at the time of writing.

We use the following techniques in our model: shifted forward measure, drift approximation, probability distribution structure exploitation, long jump, numerical integration, incomplete information handling, and calibration. Combining these techniques, the model achieves sufficient accuracy in relatively few time steps and discrete nodes, which makes it a very efficient method.

For ease of illustration, we present the lattice model based on the Trapezoidal Rule integration. A better but slightly more complicated solution is to spline the payoff functions. The cubic spline of the option payoffs can achieve higher accuracy,

especially for Greeks calculations, and higher speed. Although cubic spline takes some time, the lattice will require much fewer nodes (23 ~ 28 nodes are good enough) and can perform a much faster integration. In general, the spline method can provide a speedup factor around 3 ~ 5 times.

We have implemented the lattice model to price a variety of interest rate exotics. The algorithm can always achieve a fast convergence rate. The accuracy, however, is a bit trickier, depending on many factors: drift approximation approaches, numerical integration schemes, volatility selections, and calibration, etc. Some work, such as calibration, is more of an art than a science.

REFERENCE

Amin, K. "Jump diffusion option valuation in discrete time." Journal of Finance, Vol. 48, No. 5 (1993), pp. 1833-1863.

Brace, A., D. Gatarek, and M. Musiela. "The market model of interest rate dynamics." Mathematical Finance, Vol. 7, No. 4 (1997), pp. 127-155.

Brigo, D., and F. Mercurio. "Interest Rate Models – Theory and Practice with Smiles, Inflation and Credit." Second Edition, Springer Finance, 2006.

Das, S. "Random lattices for option pricing problems in finance." Journal of Investment Management, Vol. 9, No.2 (2011), pp. 134-152.

FinPricing, Capital market solution, https://finpricing.com/lib/IrCurve.html

Gandhi, S. and P. Hunt. "Numerical option pricing using conditioned diffusions," Mathematics of Derivative Securities, Cambridge University Press, Cambridge, 1997.

Hagan, P. "Accrual swaps and range notes." Bloomberg Technical Report, 2005.

Hull. J., and A. White. "Forward rate volatilities, swap rate volatilities and the implementation of the Libor Market Model." Journal of Fixed Income, Vol. 10, No. 2 (2000), 46-62.

Martzoukos, H., and L. Trigeorgis. "Real (investment) options with multiple sources of rare events." European Journal of Operational Research, 136 (2002), 696-706.

Piterbarg, V. "A Practitioner's guide to pricing and hedging callable LIBOR exotics in LIBOR Market Models." SSRN Working paper, 2003.

Rebonato, R. "Calibrating the BGM model." RISK, March (1999), 74-79.

APPENDIX A:

***Proof of Proposition 1*.** We rewrite (9) as

$$Y(t) = \frac{1}{\sigma_j}\left(\ln\left(\frac{F_j(t)}{F_j(0)}\right) - \frac{\sigma_j^2 t}{2}\right) \tag{A1}$$

In the general Brownian Bridge case when the Wiener process $Y(t)$ has $Y(t_1)$ =a and $Y(t_2)$ =b, the distribution of $Y(t)$ at time $t \in (t_1, t_2)$ is normal given by

$$Y(t) \sim N\left(\mu_Y(t) = a + \frac{(t-t_1)(b-a)}{(t_2-t_1)}, \quad \nu_Y(t) = \frac{(t-t_1)(t_2-t)}{(t_2-t_1)}\right) \tag{A2}$$

In our case: $t_1 = 0$, $t_2 = t$, *a*=0, *b*= $Y(t)$, $s \in (0,t)$, thus (A2) can be expressed as

$$Y(s) \sim N\left(\mu_Y(s) = \frac{s}{t} Y(t), \quad \nu_Y(s) = \frac{s(t-s)}{t} \right) \tag{A3}$$

Let $A_j(s) = \sigma_j Y(s) + \sigma_j^2 s/2$. According to the linear transformation rule, $A_j(s)$ is a normal given by

$$A_j(s) \sim \mathrm{N}\left(\mu_{Aj}(s) = \sigma_j \mu_Y(s) + \frac{\sigma_j^2 s}{2} = \frac{s}{t} \ln\left(\frac{F_j(t)}{F_j(0)} \right), \quad \nu_{Aj}(s) = \sigma_j^2 \nu_Y(s) = \frac{\sigma_j^2 s(t-s)}{t} \right) \tag{A4}$$

Let $B_j(s) = \exp\left(A_j(s)\right)$. By definition, $B_j(s)$ is a lognormal given by $B_j(s) \sim LogN\left(\mu_{Aj}(s),\ \nu_{Aj}(s)\right)$. According to the characterizations of the lognormal distribution, the mean and variance of $B_j(s)$ are

$$\mu_{Bj}(s) = E_0\left(B_j(s)\right) = \exp\left(\mu_{Aj} + \frac{\nu_{Aj}(s)}{2} \right) = \left(\frac{F_j(t)}{F_j(0)} \right)^{\frac{s}{t}} \exp\left(\frac{\sigma_j^2 s(t-s)}{2t} \right) \tag{A5a}$$

$$\nu_{Bj}(s) = \left[\exp\left(\nu_{Aj}(s)\right) - 1\right] \exp\left(2\mu_{Aj}(s) + \nu_{Aj}(s)\right) = \left(\exp\left(\frac{\sigma_j^2 s(t-s)}{t} \right) - 1 \right) \left(\frac{F_j(t)}{F_j(0)} \right)^{\frac{2s}{t}} \exp\left(\frac{\sigma_j^2 s(t-s)}{t} \right)$$

(A5b)

We have the conditional expectation of the forward rate $F_j(s)$ as

$$E\left(F_j(s)\,|_{F_j(0), F_j(t)}\right) = F_j(0) E_0\left(B_j(s)\right) = F_j(0) \left(\frac{F_j(t)}{F_j(0)} \right)^{\frac{s}{t}} \exp\left(\frac{\sigma_j^2 s(t-s)}{2t} \right) \tag{A6}$$

Proof of Proposition 2. Let $C_j(s) = 1 + \delta_j F_j(s) = 1 + \delta_j F_j(0) B_j(s)$ where $B_j(s)$ is defined above. According to the linear transformation rule, $C_j(s)$ is a lognormal given by $C_j(s) \sim LogN\left(\tilde{\mu}(s),\ \tilde{v}(s)\right)$. The mean and variance of $C_j(s)$ are

$$\mu_{Cj}(s) = 1 + \delta_j F_j(0) \mu_{Bj}(s) = 1 + \delta_j F_j(0) \left(\frac{F_j(t)}{F_j(0)} \right)^{\frac{s}{t}} \exp\left(\frac{\sigma_j^2 s(t-s)}{2t} \right) \tag{A7a}$$

$$\nu_{Cj}(s) = \delta_j^2 F_j(0)^2 \nu_{Bj}(s) = \delta_j^2 F_j(0)^2 \left(\exp\left(\frac{\sigma_j^2 s(t-s)}{t} \right) - 1 \right) \left(\frac{F_j(t)}{F_j(0)} \right)^{\frac{2s}{t}} \exp\left(\frac{\sigma_j^2 s(t-s)}{t} \right) \quad \text{(A7b)}$$

On the other hand, according to the characterizations of the lognormal distribution, the mean and variance of $C_j(s)$ are

$$\mu_{Cj}(s) = \exp\left(\hat{\mu}(s) + \frac{\hat{\nu}(s)}{2} \right) \quad \text{(A8a)}$$

$$\nu_{Cj}(s) = \left[\exp\left(\hat{\nu}(s) \right) - 1 \right] \exp\left(2\hat{\mu}(s) + \hat{\nu}(s) \right) \quad \text{(A8b)}$$

Solving the equation (A8a) and (A8b), we get

$$\hat{\mu}(s) = \ln\left(\frac{\mu_{Cj}(s)}{\sqrt{1 + \nu_{Cj}(s) / \mu_{Cj}^2(s)}} \right) \quad \text{(A9a)}$$

$$\hat{\nu}(s) = \ln\left(1 + \frac{\nu_{Cj}(s)}{\mu_{Cj}^2(s)} \right) \quad \text{(A9b)}$$

We know the first negative moment of the lognormal is $E\left(C_j^{-1}(s)\right) = \exp\left(-\hat{\mu}(s) + \hat{\nu}(s)/2 \right)$ and have the conditional expectation of the drift term as

$$\begin{aligned} E_0\left(\left. \frac{\delta_j F_j(s)}{1 + \delta_j F_j(s)} \right|_{F_j(0), F_j(t)} \right) &= E_0\left(1 - \frac{1}{1 + \delta_j F_j(s)} \right) = 1 - E_0\left(\frac{1}{C_j(s)} \right) \\ &= 1 - \exp\left(-\hat{\mu}(s) + \frac{\hat{\nu}(s)}{2} \right) = 1 - \frac{1 + \nu_{Cj}(s) / \mu_{Cj}^2(s)}{\mu_{Cj}(s)} \end{aligned} \quad \text{(A10)}$$

where $\mu_{Cj}(s)$, $\nu_{Cj}(s)$ are given by (A7a) and (A7b).

APPENDIX B:

The following pseudo-code (C++) demonstrates how to implement the model to price a callable bond. For the purpose of an easy illustration, we choose the same settings (the number of nodes and the grid space) across the lattice and use the Trapezoidal Rule for numerical integration.

```
// 2*numNodes = 2*mNumNodes = the number of nodes (S); gap = mGap = the grid space (Phi)
double priceCallableBond (BondTrade* bd, CallableBond* cb, int numNodes, double gap) {
      double pv;
      cb->fillLattice();

      // The last exercise
      CallSchedule& cs = bd->callSch[numCallSch-1];
      if (cs.term == bd->cFlow[numCashFlow-1].endDate)          // The last exercise is at maturity
            for (int i= -numNodes; i <= numNodes; i++)
                  cs.reducedValue[i+numNodes] = min (cs.callPrice,
                                                bd->cFlow[numCashFlow-1].reducedPayoff[i+numNodes]);
      else {            // The last exercise is before maturity
            for (int i= -numNodes; i <= numNodes; i++) {
                  pv = 0;
                  for (int j = bd->numCF-1; (bd->cFlow[j].endDate >= cs.term) && (j >= 0); j--) {
                        CashFlow& cf = bd->cFlow[j];
                        (cf.endDate == cs.term) ? pv += cf.reducedPayoff[i+numNodes]
                                                : pv += exp(-bondSpread*(cf.endDate-cs.term)) * cb->integral(i,
                                                      cs.vol, cf.vol, cf.endDate, cs.term, cf.reducedPayoff);
                  }
                  cs.reducedValue[i+numNodes] = min (cs.callPrice/cs.df[i+numNodes], pv);
            }
      }

      if (numCallSch > 1) {    // The remaining  exercises
            for (int i = numCallSch - 2; i>=0; i--) {
                  CallSchedule& cs = bd->callSch[i];
                  CallSchedule& preCs = bd->callSch[i+1];
                  for (int j = -numNodes; j <= numNodes; j++) {
                        pv = exp(-bondSpread * (preCs.term - cs.term))
                              * cb->integral (j, cs.vol, preCs.vol, preCs.term, cs.term, preCs.reducedValue);
                        for (int k=bd->numCF-1; k >= 0; k--)              // Count intermediate coupons
                              if ((bd->cFlow[k].endDate < preCs.term) && (bd->cFlow[k].endDate >= cs.term))
                                    pv += bd->cFlow[k].reducedPayoff[j+numNodes]
                                          * exp (-bondSpread*(bd->cFlow[k].endDate - cs.term));
                        cs.reducedValue[j+numNodes] = min (cs.callPrice/cs.df[j+numNodes], pv);
                  }
            }
      }

      // The final integral
     CallSchedule& preCs = bd->callSch[0];
     pv = cb->integral (0, 0, preCs.vol, preCs.term, 0, preCs.reducedValue) *exp(-bondSpread*(preCs.term));
     pv *= bd->cFlow[bd->numCF-1].endDf;                  // endDf: discount factor from 0 to the end date
     for (int k=bd->numCF-1; k >= 0; k--)                 // Count intermediate coupons
            if ((bd->cFlow[k].endDate < preCs.term))
                  pv += bd->cFlow[k].coupon * bd->cFlow[k].endDf * exp(-bondSpread * bd->cFlow[k].endDate);
    return pv;
}

void CallableBond::fillLattice() {
     for (int i = mTrade->numCF-1; i>=0; i--) {
            CashFlow& cf = mTrade->cFlow[i];
            if (cf.endDate < mTrade->callSch[0].term)   break;
            for (int j = -mNumNodes; j <= mNumNodes; j++)
                  fillNode(i, j, cf.startDate, mDrift);
     }
}

void CallableBond::fillNode(int cI, int nI, double vT, DriftAppx flag) {
```

```
    int numCF = mTrade->numCF;
    double avgF, expon, fwdt, drift = 0;
    CashFlow& fl = mTrade->cFlow[cI];
    if (cI == numCF-1) {      // At maturity
        fl.df[nI + mNumNodes] = 1.0;
        fl.reducedPayoff[nI + mNumNodes] = fl.notional + fl.coupon;
    }
    else if (fl.startDate <= 0)          // Starting before valuation date)
        fl.reducedPayoff[nI + mNumNodes] = fl.coupon * fl.endDf / mTrade->cFlow[numCF-1].endDf;
    else {
        fl.df[nI + mNumNodes] = 1.0;
        for (int i = numCF - 1; i > cI; i--) {
            CashFlow& cf = mTrade->cFlow[i];
            expon = (cf.vol * cf.vol * vT / 2) + cf.vol * nI * mGap;
            fwdt = cf.fwd0 * exp(-drift + expon);
            switch (flag) {          // The other cases are similar to either AAFR or CEFR
            case AAFR:  // Arithemic Average Fwd Rate
                avgF = 0.5 * (cf.fwd0 + fwdt);
                drift += vT * fl.vol * cf.vol * cf.delta * avgF / (1 + cf.delta * avgF);
                break;
            case CEFR:  // Conditional Expectation of Fwd Rate
                drift += fl.vol * cf.vol * integralFwd(cf.fwd0, fwdt, 0, vT, cf.vol, cf.delta);
                break;
            default:
                break;
            }
            fl.df[nI + mNumNodes] /= (1 + fwdt * cf.delta);      // df: discount factor maturing at maturity
        }
        fl.reducedPayoff[nI + mNumNodes] = fl.coupon / fl.df[nI + mNumNodes];
    }
}

// Gauss-Legendre integration for drift
const double xArray[] = {0, 0.1488743389, 0.4333953941, 0.6794095682, 0.8650633666, 0.9739065285};
const double wArray[] = {0, 0.2955242247, 0.2692667193, 0.2190863625, 0.1494513491, 0.0666713443};
double CallableBond::integralFwd(double F0, double Ft, double a, double b, double vol, double delta) {
    double xm = 0.5 * (b + a);
    double xr = 0.5 * (b - a);
    double ss = 0, dx = 0;
    for (int j = 1; j <= 5; j++) {
        dx = xr * xArray[j];
        ss += wArray[j] * (expectFwd(F0, Ft, (xm + dx), b, vol, delta)
            + expectFwd(F0, Ft, (xm - dx), b, vol, delta));
    }
    return ss * xr;
}

double CallableBond::expectFwd(double F0, double Ft, double s, double t, double vol, double delta) {
    double mean = F0 * pow ((Ft / F0), (s / t)) * exp(0.5 * vol * vol * s * (t - s) / t);
    return delta * mean / (1 + delta * mean);-
}

// Trapezoidal Rule Integration
double CallableBond::integral (int curPos, double curVol, double preVol, double preTerm,
                               double curTerm, double* value){
    double diffPos, tmpV, sum = 0;
    for (int k = -mNumNodes; k <= mNumNodes; k++) {
        diffPos = k*mGap - curPos*mGap + preVol * preTerm - curVol * curTerm;
        tmpV = value[k+mNumNodes] * exp (-diffPos * diffPos/(2*(preTerm - curTerm)));
        ((k == -mNumNodes) || (k == mNumNodes)) ? sum += 0.5 * tmpV : sum += tmpV;
    }
    return sum * mGap / sqrt(2 * PI * (preTerm - curTerm));
```

}

EXHIBIT 1. The Grid/Rectangular Lattice

This exhibit defines the state space for the underlying process Y_t over the first two discrete time periods. The starting state y_0 at valuation date 0 is the single root of the lattice. At each date t_i the underlying process Y_{t_i} is discretized into a number of vertical nodes/states indexed by j. The value y_{j,t_i} denotes the underlying process in state j at date t_i. The node y_{1,t_1}, for instance, can evolve to any discrete state in t_2 with certain transition probabilities. For a Brownian motion, the transition probability can be easily determined by (25).

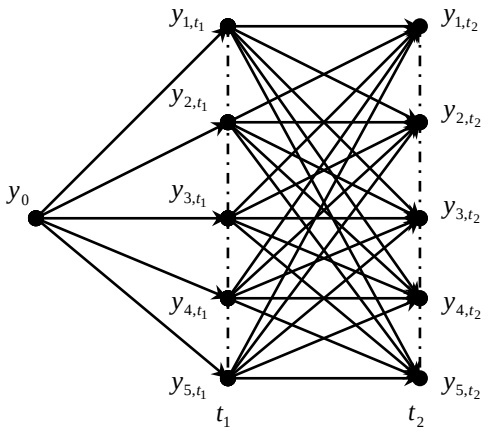


EXHIBIT 2: The Callable Bond and Associated Spot Market Data

The callable bond has a one-year maturity, a $100 principal, a quarterly payment frequency, and a 4% annual coupon rate. Delta = (end date – start date)/365 (day count: ACT/365). The discount bond $P(0,T_i)$ matures at the end date T_i. The call dates are 6, 9, and 12 months.

Cash flow index	**1**	**2**	**3**	**4**
Start date (days)	0	92	181	273
End date (days)	92 (T_1)	181 (T_2)	273 (T_3)	365 (T_4)
Delta (years)	0.252055	0.243836	0.252055	0.252055
Payoff ($)	1	1	1	101
Call Schedule (days)	-	181	273	365
Discount bond $P(0,T_i)$	0.999313	0.998557	0.997293	0.995667
Black Volatility $\hat{\sigma}_i$	-	0.337631	0.344218	0.350878

EXHIBIT 3: The LMM Lattice Structure of the Callable Bond.

The callable bond is defined in Exhibit 2. $\tilde{V}_{i,j} := \tilde{V}(T_j, y_i)$ denotes the reduced value of the callable bond at any node (*i, j*). V_1 denotes the coupon at T_1. $\bar{V}(0)$ is the value calculated by the final integration. $V(0)$ is the final callable bond value that is equal to $\bar{V}(0)$ plus the present value of V_1. The grid space is $\varphi = 0.5$ and the number of nodes is $S = 7$. This lattice has 3 steps and 7 nodes.

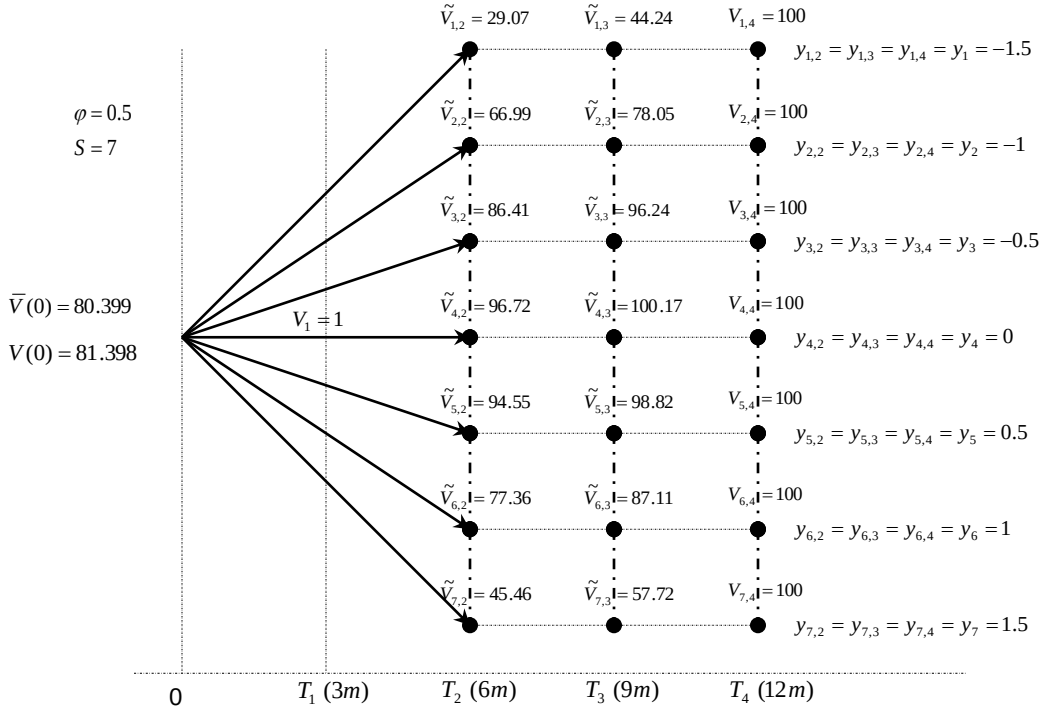


EXHIBIT 4: The Convergence Results for the Callable Bond.

The callable bond is defined in Exhibit 2. $\theta = 1$ and drift approximation is AADT. Each curve represents the convergence behavior for a given grid space (phi) as nodes are added. All calculations are converged to 100.7518.

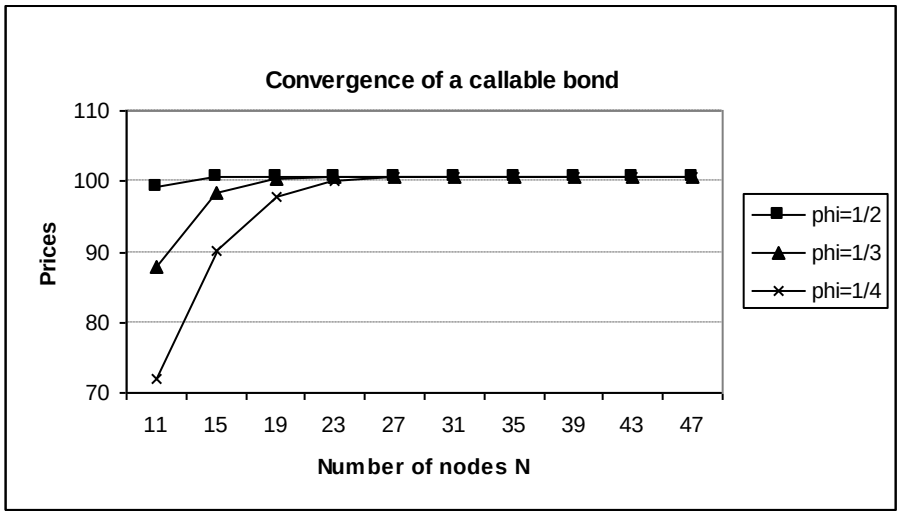


EXHIBIT 5: The Convergence Results for the Callable Capped Floater Swap

The callable capped floater swap has more than 10 years remaining in its lifetime. The floating leg has a quarterly payment frequency. The structural leg has a semi-annually payment frequency. The call schedule is semi-annual. θ =1 and drift approximation is CEDT. Each curve represents the convergence behavior for a given grid space (phi) as nodes (N) are added.

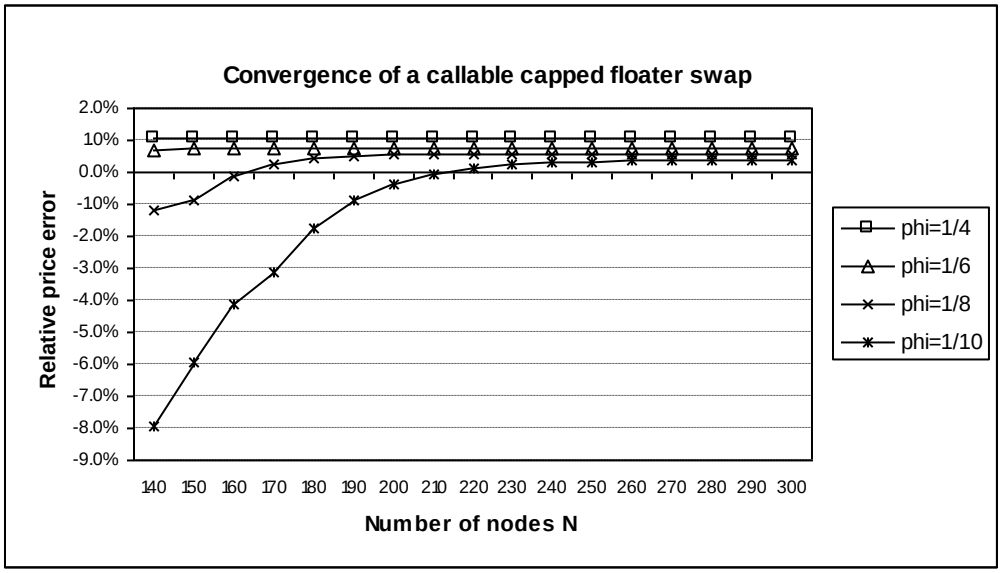


EXHIBIT 6: The Convergence Results for the Callable Range Accrual Swap

The callable range accrual swap has 10 years maturity. The floating leg has a quarterly payment frequency. The structural leg has a semi-annually payment frequency. There are 7 call opportunities. θ =1 and drift approximation is CEDT. Each curve represents the convergence behavior for a given grid space (phi) as nodes are added.

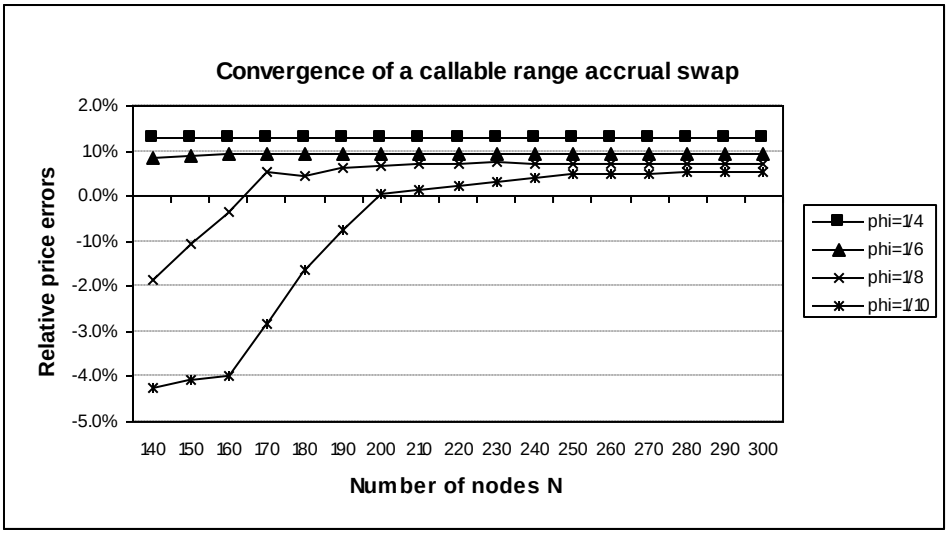


EXHIBIT 7: The Benchmark Results for the Callable Capped Floater Swap

This exhibit presents the results for model comparison. We benchmark the lattice model under different drift approximation methods with several standard market approaches, e.g., the regression-based Monte Carlo in the full LMM and the HJM trinomial tree, for both accuracy and speed. The trade is the same as the one in Exhibit 5. The grid space is φ =1/8 and the number of nodes is *S*=200. PC denotes Predictor-Corrector. The column 'Dif from MC' = 1 – (current row price) / (price of MC in LMM). All computational times are denoted in seconds on a computer with a 2.33 GHz Duo Core CPU.

Model	θ	Drift	Steps n	Calls	Nodes/Paths	Price	Err from MC	Run time
MC in LMM	-	PC	40	20	1 million	4,546,863.3	0	290.32
HJM tri-tree	-	-	1979	20	2n+1	4,602,136.3	1.22%	15.01
Our Model	1	FD	40	20	200	4,822,728.4	6.07%	0.32
	1	AAFR	40	20	200	4,637,263.2	1.99%	0.32
	1	AADT	40	20	200	4,637,718.1	2.00%	0.32
	1	GAFR	40	20	200	4,698,215.6	3.33%	0.32
	1	GADT	40	20	200	4,698,441.3	3.33%	0.32
	1	CEFR	40	20	200	4,665,210.3	2.60%	0.38

	1	CEDT	40	20	200	4,665,552.4	2.61%	0.39
	0.99	FD	40	20	200	4,708,768.9	3.56%	0.32
	0.99	AAFR	40	20	200	4,504,989.2	-0.92%	0.32
	0.99	AADT	40	20	200	4,505,426.3	-0.91%	0.32
	0.99	GAFR	40	20	200	4,609,779.5	1.38%	0.32
	0.99	GADT	40	20	200	4,609,996.6	1.39%	0.32
	0.99	CEFR	40	20	200	4,563,689.2	0.37%	0.38
	0.99	CEDT	40	20	200	4,563,730.9	0.37%	0.39

EXHIBIT 8: The Benchmark Results for the Callable Range Accrual Swap

This exhibit presents the results for model comparison. We benchmark the lattice model under different drift approximation methods with several standard market approaches, e.g., the regression-based Monte Carlo in the full LMM and the HJM trinomial tree, for both accuracy and speed. The trade is the same as the one in Exhibit 6. The grid space is $\varphi = 1/8$ and the number of nodes is S=200. The column 'Dif from MC' = 1 – (current row price) / (price of MC in LMM). All computational times are denoted in seconds on a computer with a 2.33 GHz Duo Core CPU.

Model	θ	Drift	Steps n	Calls	Nodes/Paths	Price	Dif from MC	Run time
MC in LMM	-	Euler	1801	7	1 million	585793.2	0.00%	2372.21
HJM tri-tree	-	-	1801	7	2n+1	582167.8	-0.62%	15.62
Our Model	1	FD	1801	7	200	648365.4	10.68%	0.21
	1	AAFR	1801	7	200	602482.2	2.85%	0.21
	1	AADT	1801	7	200	602742.1	2.89%	0.21
	1	GAFR	1801	7	200	616318.6	5.21%	0.21
	1	GADT	1801	7	200	616425.3	5.23%	0.21
	1	CEFR	1801	7	200	598253.3	2.13%	2.21
	1	CEDT	1801	7	200	598372.4	2.15%	2.35
	0.99	FD	1801	7	200	609373.9	4.03%	0.21
	0.99	AAFR	1801	7	200	579337.2	-1.10%	0.21
	0.99	AADT	1801	7	200	579386.3	-1.09%	0.21
	0.99	GAFR	1801	7	200	591981.5	1.06%	0.21
	0.99	GADT	1801	7	200	591917.6	1.05%	0.21
	0.99	CEFR	1801	7	200	588918.9	0.53%	2.21
	0.99	CEDT	1801	7	200	588935.7	0.54%	2.35